

《软件分析与验证》

程序验证示例



贺飞
清华大学软件学院
2024 年 3 月 1 日

1. 用一个简单示例演示程序验证的一些基本概念，包括
 - 程序规约
 - 前置条件、后置条件
 - 循环不变式
 - 秩函数
2. 更形式化的定义在后面的课程里介绍

```
int fGCD(int m, int n) {  
    int a = m;  
    int b = n;  
    while (b != 0) {  
        int tmp = a % b;  
        a = b;  
        b = tmp;  
    }  
    return a;  
}
```

? 该实现正确吗

- 先定义什么是“正确”
- “正确”需要有一个标准，称为规约。
- 如：
 - “返回 m 和 n 的最大公约数”
 - “返回 m 和 n 的最小公倍数”
 - “返回比 m 和 n 小的最大质数”
 - ...
- 如何精确无歧义地表示程序规约?

- 以前置和后置条件定义程序规约
- **前置条件**：执行代码之前假设满足的条件
- **后置条件**：执行代码之后必须满足的条件
- 解释案例。。。
- 前置和后置条件都通过逻辑表达式来定义
 - 观察一下这些逻辑表达式
 - 命题逻辑？一阶逻辑？

```
/*@ requires m >= n && n > 0;  
   ensures a = gcd(m, n)    */  
int fGCD(int m, int n) {  
    int a = m;  
    int b = n;  
    while (b != 0) {  
        int tmp = a % b;  
        a = b;  
        b = tmp;  
    }  
    return a;  
}
```

```
/*@ requires m >= n && n > 0;  
    ensures a = gcd(m, n)    */  
int fGCD(int m, int n) {  
    int a = m;  
    int b = n;  
    while (b != 0) {  
        int tmp = a % b;  
        a = b;  
        b = tmp;  
    }  
    /*@ assert a == gcd(m,n);*/  
    return a;  
}
```

- **断言**：在程序某个位置必须成立的逻辑表达式
 - 基于断言可以更灵活地定义程序规约
 - 后置条件可看作为代码块结束位置的断言
- **程序验证**：程序行为是否符合规约？即，从满足前置条件的任何状态出发，执行到指定位置是否满足断言，执行后是否满足后置条件？

```
/*@ requires m >= n && n > 0;  
    ensures a = gcd(m, n)    */  
int fGCD(int m, int n) {  
    int a = m;  
    int b = n;  
  
    while (b != 0) {  
        int tmp = a % b;  
        a = b;  
        b = tmp;  
    }  
    /*@ assert a == gcd(m,n); */  
    return a;  
}
```

- **断言**：在程序某个位置必须成立的逻辑表达式
 - 基于断言可以更灵活地定义程序规约
 - 后置条件可看作为代码块结束位置的断言
- **程序验证**：程序行为是否符合规约？即，从满足前置条件的任何状态出发，执行到指定位置是否满足断言，执行后是否满足后置条件？
 - ? 循环怎么处理
 - 程序验证的核心任务

```
/*@ requires m >= n && n > 0;
   ensures a = gcd(m, n)    */
int fGCD(int m, int n) {
    int a = m;
    int b = n;
    while (b != 0) {
        int tmp = a % b;
        a = b;
        b = tmp;
    }
    /*@ assert a == gcd(m,n); */
    return a;
}
```



From: <https://zh.wikipedia.org/wiki/>

- **定理：** 给定两个正数
 $a > b > 0$ ，如果 $b == 0$ ，则
 $\gcd(a, b) = a$ ，否则
 $\gcd(a, b) = \gcd(b, a \% b)$
- 应用该定理的辗转相除法

```
/*@ requires m >= n && n > 0;
   ensures a = gcd(m, n)    */
int fGCD(int m, int n) {
    int a = m;
    int b = n;
    while (b != 0) {
        int tmp = a % b;
        a = b;
        b = tmp;
    }
    /*@ assert a == gcd(m,n); */
    return a;
}
```

- 设 $m = 2022, n = 366$, 有

$$\begin{aligned} &GCD(2022, 366) \\ &= GCD(366, 192) \\ &= GCD(192, 174) \\ &= GCD(174, 18) \\ &= GCD(18, 12) \\ &= GCD(12, 6) \\ &= GCD(6, 0) \\ &= 6 \end{aligned}$$


```
/*@ requires m >= n && n > 0;
   ensures a = gcd(m, n)   */
int fGCD(int m, int n) {
    int a = m;
    int b = n;

    /*@ loop invariant
       a >= b && b >= 0 &&
       gcd(a,b) == gcd(m,n) */
    while (b != 0) {
        int tmp = a % b;
        a = b;
        b = tmp;
    }
    /*@ assert a == gcd(m,n); */
    return a;
}
```

- **循环不变式**：在循环头位置（无论第几次到达）永远成立的逻辑表达式：
 - 第一次到达该位置时
 $a = m, b = n$
不变式显然成立
 - 每执行完一次循环回到该位置时，根据欧几里得定理，不变式成立
 - 最后一次到达该位置时（即将退出循环），不变式仍成立，循环条件不成立
- 借助循环不变式，证明程序满足规约

- ？ 如何证明循环经过有限次迭代后一定终止
- **秩函数**：定义在程序变量上的一个算术表达式
 - 该表达式的值随着循环迭代次数的增加而严格递减，且减少幅度不会趋向于无穷小
 - 该表达式的值有下界
 - 如果找到循环的一个秩函数，则循环一定终止。
 - 为什么？

```
/*@ requires m >= n && n > 0;
    ensures a = gcd(m, n)    */
int fGCD(int m, int n) {
    int a = m;
    int b = n;
    /*@ loop invariant
        a >= b && b >= 0 &&
        gcd(a,b) == gcd(m,n) */
    /*@ loop variant a + b    */
    while (b != 0) {
        int tmp = a % b;
        a = b;
        b = tmp;
    }
    /*@ assert a == gcd(m,n);*/
    return a;
}
```

- ？ 如何证明循环经过有限次迭代后一定终止
- **秩函数**：定义在程序变量上的一个算术表达式
 - 该表达式的值随着循环迭代次数的增加而严格递减，且减少幅度不会趋向于无穷小
 - 该表达式的值有下界
 - 如果找到循环的一个秩函数，则循环一定终止。
 - 为什么？

```
/*@ requires m >= n && n > 0;
    ensures a = gcd(m, n)    */
int fGCD(int m, int n) {
    int a = m;
    int b = n;
    /*@ loop invariant
        a >= b && b >= 0 &&
        gcd(a,b) == gcd(m,n) */
    /*@ loop variant a + b    */
    while (b != 0) {
        int tmp = a % b;
        a = b;
        b = tmp;
    }
    /*@ assert a == gcd(m,n);*/
    return a;
}
```

- 设 $m = 2022, n = 366$, 表达式 $a + b$ 的值:

$$2022 + 366 = 2388$$

$$366 + 192 = 558$$

$$192 + 174 = 366$$

$$174 + 18 = 192$$

$$18 + 12 = 30$$

$$12 + 6 = 18$$

$$6 + 0 = 6$$

- 严格单调递减, 且恒大于 0
- 反证法: 假设不终止, 有限次迭代后 $a + b > 0$ 必定不成立, 与循环不变式不符

```
/*@ requires m >= n && n > 0;
    ensures a = gcd(m, n)    */
int fGCD(int m, int n) {
    int a = m;
    int b = n;
    /*@ loop invariant
        a >= b && b >= 0 &&
        gcd(a,b) == gcd(m,n) */
    /*@ loop variant a + b    */
    while (b != 0) {
        int tmp = a % b;
        a = b;
        b = tmp;
    }
    /*@ assert a == gcd(m,n);*/
    return a;
}
```

- 设 $m = 2022, n = 366$, 表达式 $a + b$ 的值:

$$2022 + 366 = 2388$$

$$366 + 192 = 558$$

$$192 + 174 = 366$$

$$174 + 18 = 192$$

$$18 + 12 = 30$$

$$12 + 6 = 18$$

$$6 + 0 = 6$$

- 严格单调递减, 且恒大于 0
- 反证法: 假设不终止, 有限次迭代后 $a + b > 0$ 必定不成立, 与循环不变式不符

```
/*@ requires m >= n && n > 0;
    ensures a = gcd(m, n)    */
int fGCD(int m, int n) {
    int a = m;
    int b = n;
    /*@ loop invariant
        a >= b && b >= 0 &&
        gcd(a,b) == gcd(m,n) */
    /*@ loop variant a + b    */
    while (b != 0) {
        int tmp = a % b;
        a = b;
        b = tmp;
    }
    /*@ assert a == gcd(m,n);*/
    return a;
}
```

- 找到循环的一个秩函数，循环一定终止！
- ？ 找到循环的一个不变式，循环一定满足规约吗
- 想象一下：
 - *true* 是否一个循环不变式？
 - 基于 *true* 能否证明程序的规约？

```
/*@ requires m >= n && n > 0;
    ensures a = gcd(m, n)    */
int fGCD(int m, int n) {
    int a = m;
    int b = n;
    /*@ loop invariant
        a >= b && b >= 0 &&
        gcd(a,b) == gcd(m,n) */
    /*@ loop variant a + b    */
    while (b != 0) {
        int tmp = a % b;
        a = b;
        b = tmp;
    }
    /*@ assert a == gcd(m,n); */
    return a;
}
```

- 逻辑基础
- 程序语义
- 霍尔证明系统
- 程序终止性
- 自动程序验证

谢谢!